| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [Interface/src](http://docs.google.com/index.html) - Logger.cpp (source / [functions](http://docs.google.com/Logger.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 34 | 59 | 57.6 % | | Date: | 2012-03-23 |  | Functions: | 14 | 15 | 93.3 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 9 | 32 | 28.1 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : #include <iostream>  2 : : #include "print.h"  3 : : #include "Logger.h"  4 : : #include "RetryFunc.h"  5 : : #include "ErrorHandler.h"  6 : :   7 : : namespace nft {  8 : :   9 : :   10 : : Logger::Logger()  11 : 1202 : :mObjLogClient(),  12 : : mLogConnected(),  13 : : mLocalLogFileName("log.txt"),  14 : : mLocalFilePointer(),  15 : : mIsLocalLogging(),  16 : 1202 : mCurrentLogLevel() {  17 : : }  18 : :   19 : 1202 : Logger::~Logger(){  20 : :   21 : : }  22 : :   23 : : void  24 : 1202 : Logger::setCurrentLogLevel(DPL::LOG\_MODE inValue) {  25 : 1202 : mCurrentLogLevel = inValue;  26 : : }  27 : :   28 : :   29 : : //Send log to server or to the local log file  30 : : void  31 : 361200 : Logger::log(const LogMessageTable::LogCode logCode, ...) {  32 : 361200 : DPL::LOG\_MODE logLevel = LogMessageTable::getLogMessage(logCode).mLevel;  33 [ + + ]: 361200 : if( logLevel > mCurrentLogLevel )  34 : : {  35 : 360082 : return;  36 : : }  37 : : //Handle the variable arguments  38 : 360082 : va\_list vargs;  39 : 360082 : va\_start(vargs,logCode);  40 : :   41 : 360082 : std::string logMessage = LogMessageTable::constructLogMessage(logCode, vargs);  42 : :   43 : 360082 : va\_end(vargs);  44 : :   45 [ + + ]: 360082 : if(!mIsLocalLogging){  46 : : //Log to logserver  47 : 1093 : mObjLogClient.sendLogData(logLevel,   48 : : const\_cast<char \*>(logMessage.c\_str()) );  49 : : }else{  50 : : //Log to local log file  51 [ + - ]: 358989 : if(mLocalFilePointer){  52 : 358989 : fprintf(mLocalFilePointer,"%s\n",logMessage.c\_str());  53 : : }  54 : : }  55 : : }  56 : :   57 : : //Returns an instance of this logger object  58 : : Logger \*  59 : : Logger::instance()  60 : 366872 : {  61 [ + + ][ + - ]: 366872 : static Logger objLogger;   62 : 366872 : return &objLogger;  63 : : }  64 : :   65 : : //start connection with the log server  66 : : void  67 : : Logger::start(DPL::LOG\_MODE logLevel,  68 : : const std::string & processName,  69 : : const std::string &logServer,  70 : 0 : const std::string &logFileName) throw(DPL::Error \*) {  71 : 0 : setLocalLogFileName( logFileName.c\_str() );  72 : 0 : mLogConnected = false;  73 [ # # ]: 0 : if(!mIsLocalLogging){  74 : : //Connect to logserver  75 : 0 : DPL::LOG\_REQ\_CONNECT logParam;  76 : 0 : logParam.logMode = logLevel;  77 : 0 : logParam.strProcName = const\_cast<char\*>(processName.c\_str());  78 : 0 : logParam.strLogServerProcName = const\_cast<char\*>(logServer.c\_str());  79 : 0 : logParam.strLogFilePath = const\_cast<char\*>(logFileName.c\_str());  80 : :   81 [ # # ]: 0 : if( 0 == mObjLogClient.initLogInfo(&logParam) ) {  82 : 0 : mLogConnected = true;  83 : : }  84 : : }  85 [ # # ][ # # ]: 0 : if( mIsLocalLogging || !mLogConnected ) {  86 : : //Open the local log file  87 : 0 : mLocalFilePointer = nftFopenWithRetry(mLocalLogFileName, "w");  88 [ # # ]: 0 : mIsLocalLogging = true;  89 : : }  90 : : }  91 : :   92 : : //Stop connection to log server  93 : : void  94 : 0 : Logger::stop() {  95 [ # # ]: 0 : if(!mIsLocalLogging){  96 : : //Disconnect to logserver  97 [ # # ]: 0 : if( true == mLogConnected )  98 : 0 : mObjLogClient.exitLogInfo();  99 : :   100 : 0 : mLogConnected = false;  101 : : }else{  102 : : //Close the local log file  103 [ # # ]: 0 : if(mLocalFilePointer){  104 : 0 : nftFcloseWithRetry(mLocalFilePointer);  105 : : }  106 : : }  107 : : }  108 : :   109 : : void  110 : 1125 : Logger::setLocalLogFileName(const char\* value) {  111 : 1125 : mLocalLogFileName = value;  112 : : }  113 : :   114 : : void  115 : 1125 : Logger::setIsLocalLogging(bool value) {  116 : 1125 : mIsLocalLogging = value;  117 : : }  118 : :   119 : : void  120 : 1125 : Logger::openOfflineLogFile() throw(DPL::Error\*) {  121 : 1125 : try  122 : : {  123 : 1125 : mLocalFilePointer = nftFopenWithRetry(mLocalLogFileName, "w");  124 : : }  125 [ # # ]: 0 : catch( DPL::Error \*pChild )  126 : : {  127 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  128 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  129 : : ErrorHandler::eErrFunction,  130 : : pErr->getErrorMsg(  131 : : ErrorHandler::eErrFunction,  132 : : "nftFopenWithRetry"));  133 : : }  134 : :   135 : : }  136 : :   137 : : void  138 : 1095 : Logger::closeOfflineLogFile() {  139 [ + - ]: 1095 : if(mLocalFilePointer)  140 : : {   141 : 1095 : try  142 : : {  143 : 1095 : nftFcloseWithRetry(mLocalFilePointer);  144 : : }  145 [ # # ]: 0 : catch(DPL::Error \*e)  146 : : {  147 : 1095 : e9k\_error("Error closing file\n");  148 : : }  149 : : }  150 : : }  151 : 1215 : } // namespace nft |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |